ЛАБОРАТОРНАЯ РАБОТА №4.
ГЛУБОКОЕ ОБУЧЕНИЕ ДЛЯ ПРОГНОЗИРОВАНИЯ ВРЕМЕННОГО РЯДА

(Продолжительность лабораторного занятия – 4 часа)

А. НАЗНАЧЕНИЕ И КРАТКАЯ ХАРАКТЕРИСТИКА РАБОТЫ

В процессе выполнения настоящей работы закрепляются знания студентов по разделам «Сверточные сети» и «Рекуррентные сети» курса «Применение методов искусственного интеллекта в электроэнергетике». Работа имеет экспериментальный характер и включает анализ данных и работы алгоритмов машинного обучения.

Целью работы является получение практических навыков работы с моделями сверточных и рекуррентных нейронных сетей для прогнозирования временных рядов в программной среде Python.

Б. СОДЕРЖАНИЕ РАБОТЫ

Работа содержит:

1. Анализ, предварительную предобработку и визуализацию данных.
2. Обучение и применение сверточной нейронной сети с оптимальными параметрами для прогнозирования временного ряда.
3. Обучение и применение рекуррентной нейронной сети с оптимальными параметрами для прогнозирования временного ряда..

Работа выполняется на компьютерах в интерактивной среде разработки JupyterLab.

В. ЗАДАНИЕ НА РАБОТУ В ЛАБОРАТОРИИ

1. Загрузить набор данных временного ряда.
2. Нормализовать временной ряд.
3. Написать функцию-генератор, формирующую окна из временного ряда (data) с заданными параметрами:

   - количество выборок в прошлом, которое определяет окно временного ряда, являющееся признаковым описанием объекта (lookback);
   - количество выборок в будущем, которое определяет окно временного ряда, являющееся целевым значением (delay);
   - два индекса массива входных данных, ограничивающих область извлечения данных (min_index, max_index);
   - параметр, определяющий, будет ли производиться формирование окон в случайном порядке или последовательно (shuffle);
   - количество окон в пакете данных (batch_size);

- шаг формирования окон из исходного временного ряда (step).

4. Инициализировать генераторы для: обучения, валидации, тестирования. Данные для валидации использовать для оценки качества модели в процессе обучения. Данные для тестирования использовать для оценки качества модели после обучения.

5. Проверить качество базового решения задачи прогнозирования без привлечения машинного обучения, предполагая, что следующее значение временного ряда равно фактическому значению предыдущего шага. Рассчитать среднюю и среднюю абсолютную ошибки. Визуализировать спрогнозированный и фактические временные ряды на одном графике.

6. Обучить многослойный перцептрон для прогнозирования временного ряда. Рассчитать среднюю и среднюю абсолютную ошибки. Визуализировать спрогнозированный и фактические временные ряды на одном графике.

7. Обучить сверточную сеть прогнозирования временного ряда. Рассчитать среднюю и среднюю абсолютную ошибки. Визуализировать спрогнозированный и фактические временные ряды на одном графике.

8. Обучить рекуррентную сеть с одним рекуррентным слоем для прогнозирования временного ряда. Рассчитать среднюю и среднюю абсолютную ошибки. Визуализировать спрогнозированный и фактические временные ряды на одном графике.

9. Повторить пункт 8 с использованием двух рекуррентных слоев.

10. Повторить пункт 9 с использованием прореживания.

Г. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К РАБОТЕ В ЛАБОРАТОРИИ

К пункту 1.

Для того, чтобы загрузить данные в формате «.csv» используйте метод read_csv библиотеки Pandas, аргументом которого является путь к файлу. Метод возвращает объект класса DataFrame.

К пункту 2.

При выполнении нормализации будьте внимательны, нормирующие коэффициенты должны быть рассчитаны по выборке, отведенной на обучение.

К пункту 3.

Пример функции-генератора, формирующей окна из временного ряда:

```
def generator(data, lookback, delay, min_index, max_index,
              shuffle = False, batch_size = 128, step = 6):
if max_index is None:
    max_index = len(data)-delay-1
  i = min_index + lookback
  while 1:
```

```
    if shuffle:
        rows = np.random.randint(min_index + lookback,
                                 max_index, size=batch_size)
    else:
        if i + batch_size >= max_index:
            i = min_index + lookback
        rows = np.arange(i, min(i + batch_size, max_index))

    i += len(rows)
    samples = np.zeros((len(rows), lookback // step,
                        data.shape[-1]))
    targets = np.zeros((len(rows),))
    for j, row in enumerate(rows):
        indices = range(rows[j] - lookback, rows[j], step)
        samples[j] = data[indices]
        targets[j] = data[rows[j] + delay] #[1]
    yield samples, targets
```

К пункту 6.

Будьте внимательны, многослойный перцептрон обучается и применяется на данных с формой тензора (размер_пакета, признаки), где признаками является каждая выборка в окне. Однако функция-генератор возвращает окна формы (размер_пакета, размер_окна, признаки), где признаками являются количество одновременно анализируемых временных рядов. Поэтому используйте слой уплощения в качестве входного:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
model = Sequential()
model.add(Flatten(input_shape=(lookback // step, data.shape[-1])))
```

К пункту 7.

Пример архитектуры сверточной сети для прогнозирования временных рядов:

```
from tensorflow.keras.layers import Flatten, Dense, Conv1D
model = Sequential()
```

```
model.add(Conv1D(filters=16, kernel_size=3, activation='relu', input_shape=(lookback // step, data.shape[-1])))
model.add(Conv1D(filters=32, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(15, activation='relu'))
model.add(Dense(1))
model.summary()
```

К пункту 8.

Пример архитектуры рекуррентной сети с одним рекуррентным слоем для прогнозирования временных рядов:

```
from tensorflow.keras.layers import GRU
model = Sequential()
model.add(GRU(32, input_shape = (None, data.shape[-1])))
model.add(Dense(1))
model.summary()
```

К пункту 9.

Пример архитектуры рекуррентной сети с двумя рекуррентными слоями для прогнозирования временных рядов:

```
model = Sequential()
model.add(GRU(32,
        return_sequences = True,
        input_shape = (None, data.shape[-1])))
model.add(GRU(32, activation  = 'relu'))
model.add(Dense(1))
model.summary()
```

К пункту 10.

Пример архитектуры рекуррентной сети с двумя рекуррентными слоями с прореживанием для прогнозирования временных рядов:

```
model = Sequential()
model.add(GRU(32,
        dropout = 0.1,
        recurrent_dropout = 0.5,
```

```
        return_sequences = True,
        input_shape = (None, data.shape[-1])))
model.add(GRU(32, activation  = 'relu',
        dropout = 0.1,
        recurrent_dropout = 0.5))
model.add(Dense(1))
model.summary()
```

Д. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ОФОРМЛЕНИЮ ИСПОЛНИТЕЛЬНОГО ОТЧЕТА

Исполнительный отчет должен включать в себя:

- титульный лист с названием лабораторной работы и фамилией студента;
- цель лабораторной работы;
- листинг кода;
- результаты работы каждого пункта задания в виде графиков Matplotlib с подписанными осями;
- выводы о проделанной работе.